UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/602,122	06/23/2003	Adam Wade Smith	13768.371	9110

47973 7590 10/03/2007

WORKMAN NYDEGGER/MICROSOFT
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UT 84111

EXAMINER	
SHIN, KYUNG H	

ART UNIT	PAPER NUMBER
2143	

MAIL DATE	DELIVERY MODE
10/03/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/602,122	SMITH ET AL.	
	Examiner	Art Unit	
	Kyung H. Shin	2143	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *23 July 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-29* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-29* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Response to Amendment

1. This action is responding to application papers filed on **7-23-2007**.

2. Claims **1 - 29** are pending. Claims **1, 2, 4, 5, 9, 10, 11, 15, 19, 20, 21, 22, 28, 29** have been amended. Claim **30** has been cancelled. Claims **1, 15, 28, 29** are independent.

Response to Arguments

3. Applicant's arguments with respect to claims 1-29 have been considered but are moot in view of the new ground(s) of rejection with Sobeski (US Patent No. 6,304,879).

3.1 Applicant argues that the referenced prior art does not disclose, usage of class (object oriented) techniques for the development of dynamic cache software. (see Remarks Pages 19-21)

The Singhal prior art discloses that the prior art invention can be implemented as a software product. (see Singhal col. 3, lines 42-44: software implementation) The Sobeski prior art discloses object oriented (languages: Java, C++, ASP.NET; programming constructs: class, method, inheritance, instance: (http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci212681,00.html)) characteristics utilized in the development of the software for the prior art invention. The Singhal and Sobeski prior art combination discloses that the prior art invention is implemented utilizing an object oriented programming language and the class data structure (base class). (see Sobeski col. 1, lines 14-16:

dynamic cache; col. 1, lines 27-32; col. 1, lines 48-50: data and methods (class, base class, inheritance) of an object; col. 5, lines 18-23; col. 5, lines 59-63: data cache objects) The developed software utilizing the base class data structure(s) can be customized to perform a plurality of cache dependency functions.

3.2 The examiner has considered the applicant's remarks concerning a system, method, and computer products for deriving custom cache dependencies. A framework that includes an extensible cache dependency base class that can be used to derive custom cache dependency classes for invalidating cache entries on some custom condition is disclosed. Applicant's arguments have thus been fully analyzed and considered but they are not persuasive.

After an additional analysis of the applicant's invention, remarks, and a search of the available prior art, it was determined that the current set of prior art consisting of Singhal (7,096,418) discloses the applicant's invention including disclosures in Remarks dated July 23, 2007.

Claim Rejections - 35 USC § 103

The text of Title 35, U.S. Code not included in this action can be found in a prior Office action.

4. Claims **1 - 29** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Singhal et al.** (US Patent No. **7,096,418**) in view of Sobeski (US Patent No. 6,304,879).

Regarding Claim 1, Singhal discloses in a server computer system that provides Web pages to requesting client computer systems, the Web pages potentially including content from locations that are external to the server computer system, the server

computer system including a cache that stores portions of cached content previously received from locations external the server computer system, a method for causing a cache entry to be dependent on a customized dependency, the method comprising the following:

c) an act of accessing a portion of content that is to be delivered to a client computer system; (see Singhal col. 3, lines 37-41: access content for client

d) an act of creating a cache entry that associates the customized dependency with the accessed portion of content; (see Singhal col. 3, lines 28-30: associate cache entry with dependent data) and

e) an act of inserting the cache entry into cache such that the validity of the cache entry is dependent on the customized dependency. (see Singhal col. 3, lines 30-33: cache entry dependent on dependency data)

Singhal discloses wherein software implementation for prior art dynamic cache invention. (see Singhal col. 3, lines 28-33: cache utilization; col. 3, lines 42-44: col. 20, lines 53-60; col. 21, lines 33-37: software, program (i.e. method, class programming)) Singhal does not specifically disclose whereby base class, customized cache dependencies can be derived.

However, Sobeski discloses:

a) an act of accessing an extensible cache dependency base class from which customized cache dependencies can be derived, the extensible cache dependency base class including a plurality of inheritable cache management

methods usable by customized dependencies derived from the extensible case dependency base class; (see Sobeski col. 1, lines 14-16: dynamic cache; col. 1, lines 27-32; col. 1, lines 48-50: data and methods (class, base class, inheritance) of an object; col. 5, lines 18-23; col. 5, lines 59-63: data cache objects)

b) an act of deriving a customized cache dependency class from the extensible cache dependency base class, the customized cache dependency class inheriting the plurality of inheritable cache management methods from the extensible case dependency base class, the customized cached dependency class also configured to implementing further unique functionality of a customized dependency customized dependency that extends the plurality of inheritable cache management methods included in the extensible cache dependency base class; (see Sobeski col. 1, lines 14-16: dynamic cache; col. 1, lines 27-32; col. 1, lines 48-50: data and methods (class, base class, inheritance) of an object; col. 5, lines 18-23; col. 5, lines 59-63: data cache objects, customized development of dependency techniques)

It would have been obvious to one of ordinary skill in the art to modify Singhal as taught by Sobeski to enable the capability for the usage of class, object-oriented programming procedures in the development of dynamic cache software. One of ordinary skill in the art would have been motivated to employ the teachings of Sobeski in order to enable the capability to utilize standard and modular programming techniques in the development of complex and sophisticated software for a dynamic cache system. (see Sobeski col. 1, lines 19-21: " *... Object-oriented*

programming environments are currently the standard environment in which computer programs are developed. ... "; col. 1, lines 23-26: " ... *Object-oriented programming environments provide a modular manner by which developers can develop complex and sophisticated computer programs. ...* ")

Regarding Claim 2, Singhal discloses the method set forth in claim 1, wherein the act of accessing an extensible cache dependency base class comprises an act of accessing a cache dependency base class that includes a notify dependency changed method such that classes derived from the cache dependency base class can implement purging functionality of the cache dependency base class by calling the notify dependency changed method. (see Singhal col. 3, lines 33-37; col. 4, line 62 - col. 5, line 1: invalidate (i.e. purge) cache content based on dependent data)

Regarding Claim 3, Singhal discloses the method set forth in claim 1, wherein the act of accessing an extensible cache dependency base class comprises an act of accessing a cache dependency base class that includes a dependency dispose method such that classes derived from the cache dependency base class can release resources occupied by a cache entry by calling the dependency dispose method. (see Singhal col. 8, lines 17-28: cache resources information updated (i.e. released))

Regarding Claim 4, Singhal discloses the method set forth in claim 1, wherein the act of deriving a customized cache dependency class from the extensible cache

dependency base class comprises an act of deriving a customized cache dependency class that can be implemented to cause a cache entry to be dependent on a database table. (see Singhal col. 3, lines 28-33: monitor cache dependency information; col. 4, lines 43-48: cache dependency, access content from database information)

Regarding Claim 5, Singhal discloses the method set forth in claim 1, wherein the act of deriving a customized cache dependency class from the extensible cache dependency base class comprises an act of deriving a customized cache dependency class that can be implemented to cause a cache entry to be dependent on a Web service. (see Singhal col. 3, lines 28-33: monitor cache dependency information; col. 4, lines 27-29: cache dependency, access information for web service (i.e. user request))

Regarding Claim 6, Singhal discloses the method as recited in claim 1, wherein the act of accessing a portion of content that is to be delivered to a client computer system comprises an act of accessing a portion of content that is to be included in a Web page. (see Singhal col. 3, lines 28-33: monitor cache dependency information; col. 4, lines 56-59: cache contents, dependency based on web page)

Regarding Claim 7, Singhal discloses the method as recited in claim 1, wherein the act of accessing a portion of content that is to be delivered to a client computer system comprises an act of accessing a portion of content from a database table. (see Singhal col. 3, lines 28-33: monitor cache dependency information; col. 4, lines 43-48: cache

dependency; access content from database information)

Regarding Claim 8, Singhal discloses the method as recited in claim 1, wherein the act of accessing a portion of content that is to be delivered to a client computer system comprises an act of accessing a portion of content from a Web service. (see Singhal col. 3, lines 28-33: monitor cache dependency information; col. 4, lines 27-29: cache dependency, access information for web service (i.e. user request))

Regarding Claim 9, Singhal discloses the method as recited in claim 1, wherein the act of creating a cache entry that associates the customized dependency with the accessed portion of content comprises an act of creating a cache entry that causes content from a database table to be dependent on the database table. (see Singhal col. 3, lines 28-33: monitor cache dependency information; col. 4, lines 43-48: cache dependency, create content based on database information)

Regarding Claim 10, Singhal discloses the method as recited in claim 1, wherein the act of creating a cache entry that associates the customized dependency with the accessed portion of content comprises an act of creating a cache entry that causes content from a database table to be dependent on an aggregate dependency. (see Singhal col. 3, lines 28-33: monitor cache dependency information; col. 4, lines 41-43: static content, dynamic content, multiple types of dependencies)

Regarding Claim 11, Singhal discloses the method as recited in claim 1, wherein the act of creating a cache entry that associates the customized dependency with the accessed portion of content comprises an act of creating a cache entry that causes content from a Web service to be dependent on the Web service. (see Singhal col. 3, lines 28-33: monitor cache dependency information; col. 4, lines 27-29: cache dependency based on web service (i.e. user request) information)

Regarding Claim 12, Singhal discloses the method as recited in claim 1, wherein the act of inserting the cache entry into cache such that the validity of the cache entry is dependent on the customized dependency comprises an act of inserting a cache entry that is dependent on a database table. (see Singhal col. 3, lines 28-33: monitor cache dependency information; col. 4, lines 43-48: cache dependency based on database content information)

Regarding Claim 13, Singhal discloses the method as recited in claim 1, wherein the act of inserting the cache entry into cache such that the validity of the cache entry is dependent on the customized dependency comprises an act of inserting a cache entry that is dependent on an aggregate dependency. (see Singhal col. 3, lines 28-33: monitor cache dependency information; col. 4, lines 41-43: static content, dynamic content, multiple types of dependencies)

Regarding Claim 14, Singhal discloses the method set forth in claim 1, wherein a start

time property does not need to be specified to invoke the functionality of the extensible cache dependency base class. (see Singhal col. 3, lines 28-33: cache dependency system; col. 21, lines 18-22: processor; col. 20, lines 53-60; col. 21, lines 33-37; col. 21, line 61 - col. 22, line 3: software, computer readable medium, program, (i.e. class, method software development))

Regarding Claim 15, Singhal discloses in a server computer system that provides Web pages to requesting client computer systems, the Web pages potentially including content from locations that are external to the server computer system, the server computer system including a cache that stores portions of cached content previously received from locations external the server computer system, a method for purging a cache entry, the method comprising the following:

b) an act of determining if the one or more custom dependency conditions have been satisfied; (see Singhal col. 3, lines 30-33: determine if cache contents changed)

c) an act of receiving an indication that the one or more custom dependency conditions have been satisfied; (see Singhal col. 3, lines 30-33: receive indication of dependency change) and

d) an act of purging a cache entry at the server computer system in response to receiving the indication that the one or more custom dependency conditions have been satisfied. (see Singhal col. 3, lines 33-37: invalidate cache contents reloaded)

Singhal discloses wherein an act of monitoring one or more custom dependency conditions, and software implementation for prior art dynamic cache invention. (see Singhal col. 3, lines 28-30: monitor dependency condition; col. 3, lines 28-33: cache utilization; col. 3, lines 42-44: col. 20, lines 53-60; col. 21, lines 33-37: software, program (i.e. method, class programming)) Singhal does not specifically disclose whereby base class, customized cache dependencies can be derived.

However, Sobeski discloses:

a) one or more custom dependency conditions associated with an instance of a customized cache dependency that extends a plurality of cache management methods inherited from an extensible cache dependency base class to implement unique functionality, the customized cache dependency corresponding to a customized cache dependency class that was derived from the extensible cache dependency base class; (see Sobeski col. 1, lines 14-16: dynamic cache; col. 1, lines 27-32; col. 1, lines 48-50: data and methods (class, base class, inheritance) of an object; col. 5, lines 18-23; col. 5, lines 59-63: data cache objects)

It would have been obvious to one of ordinary skill in the art to modify Singhal as taught by Sobeski to enable the capability for the usage of class, object-oriented programming procedures in the development of dynamic cache software. One of ordinary skill in the art would have been motivated to employ the teachings of Sobeski in order to enable the capability to utilize standard and modular

programming techniques in the development of complex and sophisticated software for a dynamic cache system. (see Sobeski col. 1, lines 19-21; col. 1, lines 23-26)

Regarding Claim 16, Singhal discloses the method as recited in claim 15, wherein the act of monitoring one or more custom dependency conditions associated with an instance of a customized cache dependency comprises an act of monitoring a database table dependency. (see Singhal col. 3, lines 28-33: monitor cache dependency; col. 4, lines 43-48: cache contents, dependency based on database content)

Regarding Claim 17, Singhal discloses the method as recited in claim 15, wherein the act of monitoring one or more custom dependency conditions associated with an instance of a customized cache dependency comprises an act of monitoring a Web services dependency. (see Singhal col. 2, lines 28-33: monitor cache dependency; col. 4, lines 27-29: web service)

Regarding Claim 18, Singhal discloses the method as recited in claim 15, wherein the act of monitoring one or more custom dependency conditions associated with an instance of a customized cache dependency comprises an act of monitoring an aggregate dependency. (see Singhal col. 3, lines 28-30: monitor cache dependency information; col. 4, lines 41-43: cache dependency based on aggregated dependencies, static and dynamic cache content)

Regarding Claim 19, Singhal discloses the method as recited in claim 15, wherein the act of determining if the one or more custom dependency conditions have been be satisfied comprises an act of determining if a flag entry has been incremented. (see Singhal col. 3, lines 30-33: change (i.e. incremented) in dependency value (i.e. flag value) or attribute)

Regarding Claim 20, Singhal discloses the method as recited in claim 15, wherein the act of determining if the or more custom dependency conditions have been satisfied comprises an act of determining if a dependency condition associated with an aggregate dependency has been satisfied. (see Singhal col. 3, lines 30-37: dependency condition processed; col. 4, lines 41-43: cache dependency based on aggregated dependencies, static and dynamic cache content)

Regarding Claim 21, Singhal discloses the method as recited in claim 15, wherein the act of determining if the one or more custom dependency conditions have been satisfied comprises calling a notify dependency changed method upon the happening of some event specified by an instance of a customized cache dependency class. (see Singhal col. 3, lines 33-37: cache dependency, event processing completed)

Regarding Claim 22, Singhal discloses the method as recited in claim 15, wherein the act of receiving an indication that the one or more custom dependency conditions have been satisfied comprises an act of receiving an indication from a notify dependency

changed method that the one or more custom dependency conditions have been satisfied. (see Singhal col. 3, lines 33-37: receive information concerning event condition to affect cache dependency)

Regarding Claim 23, Singhal discloses the method as recited in claim 15, wherein the act of purging a cache entry at the server computer system in response to receiving the indication comprises an act of purging a cache entry that was dependent on a database table. (see Singhal col. 3, lines 30-37: invalidate (i.e. purge) cache content due to change in dependency value; col. 4, lines 43-48: cache dependency, database contents)

Regarding Claim 24, Singhal discloses the method as recited in claim 15, wherein the act of purging a cache entry at the server computer system in response to receiving the indication comprises an act of purging a cache entry that was dependent on a Web service. (see Singhal col. 3, lines 30-37: invalidate (i.e. purge) cache content due to change in dependency value; col. 4, lines 27-29: web service (i.e. user request))

Regarding Claim 25, Singhal discloses the method as recited in claim 15, wherein the act of purging a cache entry at the server computer system in response to receiving the indication comprises an act of purging a cache entry that was dependent on an aggregate dependency. (see Singhal col. 3, lines 30-37: invalidate (i.e. purge) cache content due to change in dependency value; col. 4, lines 41-43: aggregate dependency,

static and dynamic content)

Regarding Claim 26, Singhal discloses the method as recited in claim 15, further comprising: an act of releasing resources that were consumed to maintain the purged cached entry. (see Singhal col. 8, lines 17-28: resource release, update indices (i.e. parameters) designating cache dependency information)

Regarding Claim 27, Singhal discloses the method as recited in claim 24, wherein the act of releasing resources that were consumed to maintain the purged cached entry comprises an act of calling a DependencyDispose method associated with customized cache dependency, the customized cache dependency being derived from an extensible cache dependency base class. (see Singhal col. 3, lines 33-37: invalidate (i.e. purge) cache contents based on cache dependency information; col. 21, lines 18-22: processor; col. 20, lines 53-60; col. 21, lines 33-37; col. 21, line 61 - col. 22, line 3: software, computer readable medium, program, (i.e. class, method software development))

Regarding Claim 28, Singhal discloses a computer program product for use in a server computer system that provides Web pages to requesting client computer systems, the Web pages potentially including content from locations that are external to the server computer system, the server computer system including a cache that stores portions of cached content previously received from locations external to the server computer

system, the computer program product for implementing a method for causing a cache entry to be dependent on a customized dependency, the computer program product comprising one or more computer-readable media having stored thereon computer executable instructions that, when executed by a processor, cause the server computer system to perform the following:

c) access a portion of content that is to be delivered to a client computer system; (see Singhal col. 3, lines 37-41: access content)

d) create a cache entry that associates the customized dependency with the accessed portion of content; (see Singhal col. 3, lines 30-33:) and

e) insert the cache entry into cache such that the validity of the cache entry is dependent on the customized dependency. (see Singhal col. 3, lines 28-33: cache entry dependent on data attributes)

Singhal discloses wherein software implementation for prior art dynamic cache invention. (see Singhal col. 3, lines 28-33: cache utilization; col. 3, lines 28-33: cache dependency capability; col. 21, lines 18-22: processor col. 3, lines 42-44: col. 20, lines 53-60; col. 21, lines 33-37: software, program (i.e. method, class programming)); col. 20, lines 53-60; col. 21, lines 33-37; col. 21, line 61 - col. 22, line 3: software, computer readable medium, program) Singhal does not specifically disclose whereby base class, customized cache dependencies can be derived. However, Sobeski discloses:

a) access an extensible cache dependency base class from which customized cache dependencies can be derived, the extensible cache dependency base class including a plurality of inheritable cache management methods usable by customized dependencies derived from the extensible case dependency base class; (see Sobeski col. 1, lines 14-16: dynamic cache; col. 1, lines 27-32; col. 1, lines 48-50: data and methods (class, base class, inheritance) of an object; col. 5, lines 18-23; col. 5, lines 59-63: data cache objects)

b) derive a customized cache dependency class from the extensible cache dependency base class, the customized cache dependency class inheriting the plurality of inheritable cache management methods from the extensible case dependency base class, the customized cache dependency class also configured to implementing further unique functionality of a customized dependency customized dependency that extends the plurality of inheritable cache management methods included in the extensible cache dependency base class; (see Sobeski col. 1, lines 14-16: dynamic cache; col. 1, lines 27-32; col. 1, lines 48-50: data and methods (class, base class, inheritance) of an object; col. 5, lines 18-23; col. 5, lines 59-63: data cache objects)

It would have been obvious to one of ordinary skill in the art to modify Singhal as taught by Sobeski to enable the capability for the usage of class, object-oriented programming procedures in the development of dynamic cache software. One of ordinary skill in the art would have been motivated to employ the teachings of Sobeski in order to enable the capability to utilize standard and modular

programming techniques in the development of complex and sophisticated software for a dynamic cache system. (see Sobeski col. 1, lines 19-21; col. 1, lines 23-26)

Regarding Claim 29, Singhal discloses a computer program product for use in a server computer system that provides Web pages to requesting client computer systems, the Web pages potentially including content from locations that are external to the server computer system, the server computer system including a cache that stores portions of cached content previously received from locations external the server computer system, the computer program product for implementing a method for purging a cache entry, the computer program product comprising one or more computer-readable media having stored thereon computer executable instructions that, when executed by a processor, cause the server computer system to perform the following:

b) determine if the one or more custom dependency conditions have been be satisfied; (see Singhal col. 3, lines 30-33: determine cache dependency value)

c) receive an indication that the one or more custom dependency conditions have been satisfied; (see Singhal col. 3, lines 33-37: determine cache dependency value) and

d) purge a cache entry at the server computer system in response to receiving the indication that the one or more custom dependency conditions have been satisfied. (see Singhal col. 3, lines 33-37: invalidate (i.e. purge) cache dependency entry)

Singhal discloses wherein software implementation for prior art dynamic cache invention. (see Singhal col. 3, lines 28-33: cache utilization, monitor cache dependency information; col. 3, lines 42-44: col. 20, lines 53-60; col. 21, lines 33-37: software, program (i.e. method, instance of a class, programming)) Singhal does not specifically disclose whereby base class, customized cache dependencies can be derived.

However, Sobeski discloses:

a) one or more custom dependency conditions associated with an instance of a customized cache dependency that extends a plurality of cache management methods inherited from an extensible cache dependency base class to implement unique functionality, the customized cache dependency corresponding to a customized cache dependency class that was derived from the extensible cache dependency base class; (see Sobeski col. 1, lines 14-16: dynamic cache; col. 1, lines 27-32; col. 1, lines 48-50: data and methods (class, base class, inheritance) of an object; col. 5, lines 18-23; col. 5, lines 59-63: data cache objects)

It would have been obvious to one of ordinary skill in the art to modify Singhal as taught by Sobeski to enable the capability for the usage of class, object-oriented programming procedures in the development of dynamic cache software. One of ordinary skill in the art would have been motivated to employ the teachings of Sobeski in order to enable the capability to utilize standard and modular

programming techniques in the development of complex and sophisticated software for a dynamic cache system. (see Sobeski col. 1, lines 19-21; col. 1, lines 23-26)

Conclusion

5. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Kyung H. Shin whose telephone number is (571) 272-3920. The examiner can normally be reached on 9:30 am - 6 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, David A. Wiley can be reached on (571) 272-3923. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

K H S
Kyung H Shin
Patent Examiner
Art Unit 2143

KHS
September 17, 2007

DAVID WILEY
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100